VAWKUM Transactions on Computer Sciences
http://vfast.org/journals/index.php/VTCS@ 2018, ISSN(e):2308-8168, ISSN(p): 2411-6335
Volume 15, Number 2, May-August , 2018 *pp:59-65*

EFFECTS OF REFACTORING AND DESIGN PATTERNS ON THE SOFTWARE SOURCE CODE QUALITY : AN EMPIRICAL ASSESSMENT

MUHAMMAD SHAHID RAEES[1], MUHAMMAD ADEEL ASHRAF[2]

`

[1]National University of Computer and Emerging Sciences NUCES-FAST, Lahore, Punjab, Pakistan
[2]Department of Computer Science, University of Management and Technology, Lahore, Punjab, Pakistan
Email: L165090@lhr.nu.edu.pk

ABSTRACT. *The purpose of this research is to identify the impact of refactoring techniques and design patterns on the source code of the Data management application. The research methodology used for this research is first identify the source code bad smells and then remove them by apply refactoring techniques and design patterns. To evaluate the impact of refactoring and design patterns on the source code quality; the code metrics and visual studio 2015 community edition is used to evaluate code metrics. The results of this research shows that the applying refactoring and design patterns in a combination has a positive effect on the source code quality. The code reusability, expandability and understandability will be increased.*

Keywords*:* Design Patterns; Refactoring; Code Metrics

1. **Introduction.** The purpose or objective of this study is to identify the effects of refactoring and design patterns on the source code quality of Data Management application before and after applying refactoring and Design patterns. Does the source code quality improves or not? And the design patterns shall be used with thoughtfulness because they may delay the maintenance and the evaluation [6] of the source code.

The application's source code goes under this research study is named as "Data Management Application" it is a desktop base windows form application and built for in-house and server base content management. It contains different modules that are create questions, answers, explanation import export questions data on server, export data into different format (.CSV, .RTF, .SQL). Technologies used to build data management application are C#, .Net Framework 4.5.2, SQL Database.

The design patterns are used in this study are strategy, Abstract Factory, and Façade design patterns. The three quality attributes reusability, expandability and understandability used in this research study to determine the effect of design patterns on software code quality. Reusability the degree to which a module, source code or design can be reused this is an important attribute while expanding the project with the combination of existing and new requirements, expandability the degree to which a design can be extended and understandability the degree to which the source code of a project can be easily under stood by the code reviewer of maintance personnel it is also an important attribute considering source code maintance and expandability of the source code.

There are five code metrics are used to evaluate the result of this research which are maintainability Index, Cyclometic Complexity, Depth of Inheritance, Class Coupling, Line of Code. Microsoft Visual Studio 2015 community edition is used to evaluate code metrics. There are three hypotheses are presented in this research study that might be proved or disapproved after this research study.

The methodology of this research is divided into four steps. The first step is to identify the code bad smell form the source code of the application. The second step is to apply the refactoring to the code. The third step is to apply design pattern to the code. And the fourth and final step of this research is to evaluate the effects of this research by calculating the code metrics.

The section I introduction contains the research question and summary of the proposal what is require: The Source Code Quality improves after applying design patterns, why does it require: it requires some quality attributes to justify the findings of this research study, and how it will be done: It will be done applying design

patterns (strategy, abstract factory, and composite design patterns) on the source code of data management application. Section 2 present contains literature on topics, methods, theoretical approach Literature review section focus on what and how part of the problem what to study and how to. Section 3 focus on Proposed Methodology, research design, procedure, and hypotheses. Section 4 present the results of the research. Section 5 Conclude the research.

2. Literature Review. Refactoring is a process of changing the code in such a way the behavior of the code does not change yet internal structured of the code is modified [8]. A design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different states [9].

Cyclomatic complexity is a code complexity measurement that is being associated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module [7].

Maintainability Index is a software metric which measures how easy to support and change the source code is. The maintainability index is calculated as a factored formula consisting of Lines of Code, Cyclomatic Complexity and Halstead Volume. It is used in several automated software metric tools, including the Microsoft Visual Studio. [10]

Depth of Inheritance The number of class definitions that extend to the root of the class hierarchy. It is more difficult to understand a method and fields that are defined or redefined when class hierarchies are deeper. [12]. Class Coupling the Class coupling is how many classes are directly depended upon a class. It is good to implement less dependent architecture. [12]. Lines of Code (LOC) is the approximate number of lines in the source code. The count is based on the Intermediate Language (IL) code and this is therefore not the exact number of lines in the source code file. [12]. Since the design patterns are introduction by Gamma et al. [1] in 1994, there has been an increasing consideration on the usage of the design patterns. Their research present some lines of work on the effect of patterns on code quality. Lange and Nakamura established [2] that design patterns can help as guide in program exploration and thus make the process of program understanding more efficient. However, this study was limited to a single quality attribute and to a little number of patterns.

Wydaeghe et al. [3] presented a study on the actual use of six design patterns. They discussed the effect of these patterns on reusability, modularity, flexibility, and understand ability, and the difficulty to concretely implement these patterns. They concluded that not all patterns have a positive effect on quality traits. Yet, this study was limited to the authors' own experience and it can hardly have been generalized to other contexts of development. Wendorff [4] evaluated the use of design patterns in large commercial systems and concluded that patterns do not necessarily improve their design. Indeed, a design can be over-engineered [5] and the cost of re-moving patterns is high. He did not perform a study on the impact of patterns on quality and provided only qualitative arguments.

3. Proposed Methodology. The methodology for this research is divided into four steps. The first step is to identify the code bad smell form the source code of the application. The second step is to apply the refactoring techniques to the source code to remove the bad smell from the code. The third step is to apply design pattern to the source code. And the fourth and final step of this research is to evaluate the effects of this research by calculating the code metrics.

There are three hypotheses at given at this stage. The research shows that these are valid or not after conducting this research work. Hypotheses are:

H1o: MI will not change significantly after applying Refactoring and Design Patterns on to the source code.

H11: MI will increase significantly after applying Refactoring and Design Patterns on to the source code.

H2o: Client code will not significantly affected after applying Refactoring and Design Patterns on to the source code.

H21: Client code will significantly affected after applying Refactoring and Design Patterns on to the source code.

H3o: Source code reusability will not increase significantly after applying Refactoring and

Design Patterns on to the source code.

H31: Source code reusability will increase significantly after applying Refactoring and Design Patterns on to the source code.

Table I represent the identified bad smell from the source code of each class and Table II represent the proposed refactoring techniques and Design patterns applied on to the source code.

These code bad smells are identified manually exploring the code and refactoring techniques are proposed based on the code bad smells which are identified.

The QuestionDetailForm.cs and ExportDataForm.cs are the god classes and have very long methods, Duplicate code and it was very difficult to manage them.

TABLE I.

Class	Bad Smells
QuestionDetailForm.cs	• Duplicate Code • Feature Envy • Divergent Changes • Long method
ExportDataForm.cs	• Duplicate Code • Large Methods • Temporary Fields • Middle Man • Feature Envy • Long Method • Divergent Changes
ImportExternalDBForm.cs	• Duplicate Code • Large Methods • Temporary Fields • Middle Man • Feature Envy • Long Method • Divergent Changes

a. Existing Source Code bad smell

TABLE II.

Class	Refactoring	Design Pattern
ParseDataHandler.cs	• New Class	• Facad
CSVHandler.cs	• New Class	• Streategy
BLLQuestions.cs	• New Class	• Strategy • Abstract Factory
QuestionDetailForm.cs	• Extract Method • Extract Class • Replace Temp with Query • Replace Method with Method object • Decompose Conditions	• Strategy • Abstract Factory • Facad

ExportDataForm.cs	• Extract Method • Pull Up Method • Replace Temp with Query • Replace Method with Method object • Decompose Conditions • Move method	• Strategy • Abstract Factory • Facad
ImportExternalDBForm.cs	• Extract Method • Pull Up Method • Replace Temp with Query • Replace Method with Method object • Decompose Conditions • Move method	• Strategy • Abstract Factory • Facad

b. Source Code after Applying refactoring and Design Patterns.

A. Formula and Equation

The formula used in this research study are:

1. Maintainability Index (MI):

Calculation formula used by Microsoft Visual Studio Science 2008 and uses the shifted scale 0 to 100 derivative

MI = MAX (0, (171 - 5.2 * ln(V) - 0.23 * (M) - 16.2 * ln(LOC)) *100 / 171)

Explanation:

V = *Halstead Volume*

M = *Cyclomatic Complexity*

LOC = *Lines of Code*

ln = *Natural Logarithms*

The major factor in MI is LOC. [13]

2. Cyclomatic Complexity (CCM)

M = E-N+2P

Explanation:

M = *Cyclomatic Complexity*

E = *No. of edges of a graph*

N = *No. of nodes of a graph*

P = *No. of connected components or the No. of nodes that have exist points*. [11]

3. Depth of Inheritance (DOI)

The number of class definitions that extend to the root of the class hierarchy.

4. *Class Coupling (CC)*

The Class coupling is how many classes are directly depended upon a class.

5. Lines of Code (LOC)

LOC is the approximate number of lines in the source code. The count is based on the Intermediate

Language (IL) code and this is therefore not the exact number of lines in the source code file.

4. Discussion of the Results: The results of software code metrics are represented in numeric form. These code metrics results are calculated by using visual studio 2015 community edition tool.

The table III shows the results of existing source code structure of Data Management application and code metrics of the three classes (QuestionDetailForm.cs, ExportsData-From.cs, and ImportExternalDBForm.cs). The table structure is that it contains 5 columns and 4 rows. The top row contains the headings and rest of the four contains the results of code metrics. These code metrics are the collected before applying refactoring and design patterns on to the source code. The number shows that the code is poorly structured and has less maintainability which shows that it is very difficult to maintain the source code. And that was the initial motivation to this research.

TABLE III.

Class	***MI***	***CCM***	***DOI***	***CC***	***LOC***
QuestionDetailForm.cs	47	1083	10	205	3793
ExportDataForm.cs	42	668	9	147	1691
ImportExternalDBForm.cs	31	327	9	114	1681

a. Code Structure and Code Metrics Before applying refactoring Design patterns

The table II shows the results of the source code structure of Data Management application and code metrics of the six classes (, QuestionDetailForm.cs (A), ExportsData-From.cs (B), ImportExternalDBForm.cs (C) ParseDataHandl-er.cs(D), CSVHandler.cs (E), and BLLQuestions.cs (F)) after applying the refactoring and designs patterns. The table IV structure is that it contains 5 columns and 7 rows. The top row contains the headings and rest of the four contains the results of code metrics.

TABLE IV.

Class	***MI***	***CCM***	***DOI***	***CC***	***LOC***
QuestionDetailForm.cs	46	386	9	124	2138
ExportDataForm.cs	54	247	9	113	808
ImportExternalDBForm.cs	39	199	9	87	1060
ParseDataHandler.cs	72	29	1	33	94
CSVHandler.cs	65	11	1	6	18
BLLQuestions.cs	42	18	1	21	65

b. Code Structure and Code Metrics After applying refactoring Design patterns

There are three new classes (D, E, and F) are constructed as a result of removing the identified code bad smells and applying refactoring techniques and design patterns on the code. The MI class A is significantly decreased because of DOI is increased by applying design patterns (strategy, facad and, abstract factory) in a combination, and CCM, CC, and LOC is decreased is due to applying, duplicate code, large methods, temporary fields, feature envy, long Method, divergent changes refactoring techniques.

The MI of class B is increased 28% (from 42 to 54) which good due to decreasing in LOC 38% (from 1691 to 1060), CMM 63% (from 668 to 247) and CC 23.12 %(from147 to 113). The MI of class C is increased 25% (from 31 to 39) which good due to decreasing in LOC 38% (from 1691 to 1060), CMM 40% (from 327 to 199) and CC 24 %(from114 to 87). These significant changes in code metrics are by applying, long Method, duplicate code, temporary fields, large methods, long Method, feature envy, divergent changes refactoring techniques.

These results after applying refactoring and design patterns shows that the code structure of Data Management application is change new class (D, E, F) are added into the source code and the Maintainability index is increased for the majority of the classes code which is good for code maintainability and

understandability. Cyclomatic Complexity is decrease which good for code understandability. Depth of inheritance is also decreased code coupling and LOC are also decreased that is better for cod reusability, expandability, and understandability.

This research shows that the applying refactoring and design patterns in combination results on the source code has a positive impact or effect. The results justify the initially provided H11, H2o, H31 hypotheses which were given at the earlier stage of this research study.

Conclusion. This research study is conducted to find either the code quality improves or not after applying the deign patterns and applying design patterns with thoughtfulness because they may delay the maintenance and the evaluation. The major factor in MI and the higher value of maintainability index presents that the Maintainability of the source code is easy and lower value indicated the harder to maintain the source code. Cyclomatic Complexity is the source code complexity measurement and is correlated to source code errors. The lower Cyclomatic complexity of a program tends to easier the understandability and lower the risk to modify the program. Depth of Inheritance The number of class definitions that extend to the root of the class hierarchy. It is more difficult to understand a method and fields that are defined or redefined when class hierarchies are deeper. Class Coupling the Class coupling is how many classes are directly depended upon a class. It is good to implement less dependent architecture. Lines of Code (LOC) is the approximate number of lines in the source code. The count is based on the Intermediate Language (IL) code and this is therefore not the exact number of lines in the source code file. This research results after applying refactoring and design patterns show that the Maintainability index is increased for majority of the classes code which is good for code maintainability and understandability. Cyclomatic Complexity is decrease which good for code understandability. Depth of inheritance is also decreased code coupling and LOC are also decreased that is better for cod reusability, expandability, and understandability. The hypothesis H11, H2o, H31 are satisfied. This research shows that the applying refactoring and design patterns on the source code has a positive impact or effect.

REFERENCES

[1] Gamma, E., Helm, R., Johnson, R., Vlissides, J., & Patterns, D. (1995). Elements of reusable object-oriented software. Reading: Addison-Wesley.

[2] Lange, D. B., & Nakamura, Y. (1995, October). Interactive visualization of design patterns can help in framework understanding. In ACM Sigplan Notices (Vol. 30, No. 10, pp. 342-357). ACM.

[3] Advani, D., Hassoun, Y., & Counsell, S. (2006, April). Extracting refactoring trends from open-source software and a possible solution to the'related refactoring'conundrum. In Proceedings of the 2006 ACM symposium on Applied computing (pp. 1713-1720). ACM.

[4] Wendorff, P. (2001). Assessment of design patterns during software reengineering: Lessons learned from a large commercial project. In Software Maintenance and Reengineering, 2001. Fifth European Conference on (pp. 77-84). IEEE.

[5] Xing, Z., & Stroulia, E. (2006, October). Refactoring detection based on umldiff change-facts queries. In null (pp. 263-274). IEEE.

[6] Khomh, F., & Gueheneuce, Y. G. (2008, April). Do design patterns impact software quality positively?. In Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on (pp. 274-278). IEEE.

[7] Tran, T. Software Complexity Model.

[8] Bozhikova, V., Stoeva, M., Georgiev, B., & Nikolaeva, D. (2017, September). Improving the software quality—an educational approach. In Scientific Conference Electronics (ET), 2017 XXVI International (pp. 1-4). IEEE.

[9] Sarcar, V. (2016). FAQ. In Java Design Patterns (pp. 163-167). Apress, Berkeley, CA.

[10] Najm, N. M. A. M. (2014). Measuring Maintainability Index of a Software Depending on Line of Code Only. vol, 16, 64-69.

[11] Tran, T. Software Complexity Model, https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2017/systems/Wednesday/Track6/19745_Tran.pdf

[12] ZHANG, Y., & ZHENG, W. (2009). http://msdn. microsoft. com/en-us/library/ms783323 (VS. 85). aspx http://msdn. microsoft. com/en-us/library/ms783323 (VS. 85). aspx. IEICE transactions on information and systems, 92(12), 2422-2429.

[13] Najm, N. M. A. M. (2014). Measuring Maintainability Index of a Software Depending on Line of Code Only. vol, 16, 64-69.

I. APPENDICES

A. Appendix A: Existing Code Metrics

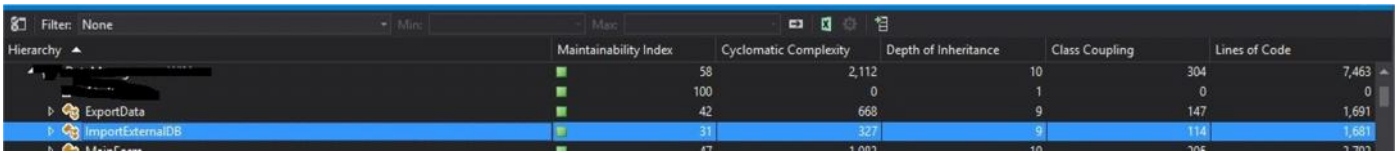

Figure 1: Import and Export Data

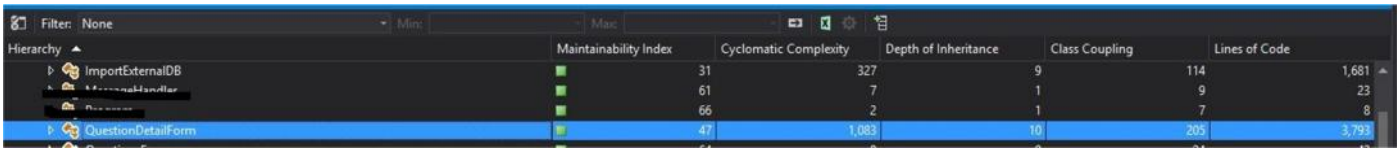

Figure 2: Question Detail Form

B. Appendix B: Code Metrics After Appling Refactoring and Design patterns

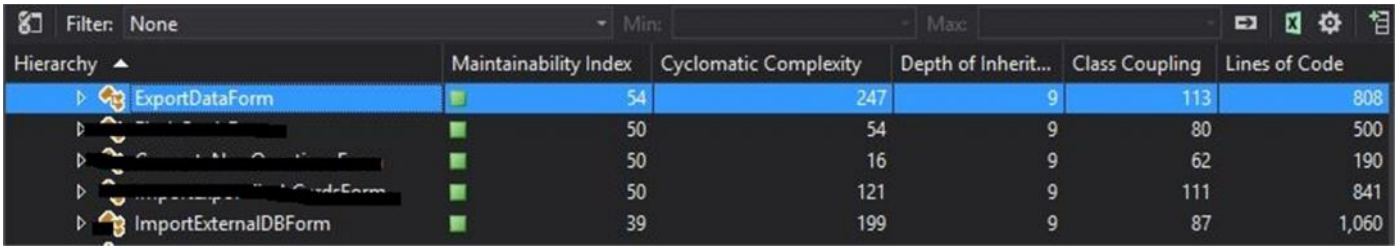

Figure 3: Import and Export Data

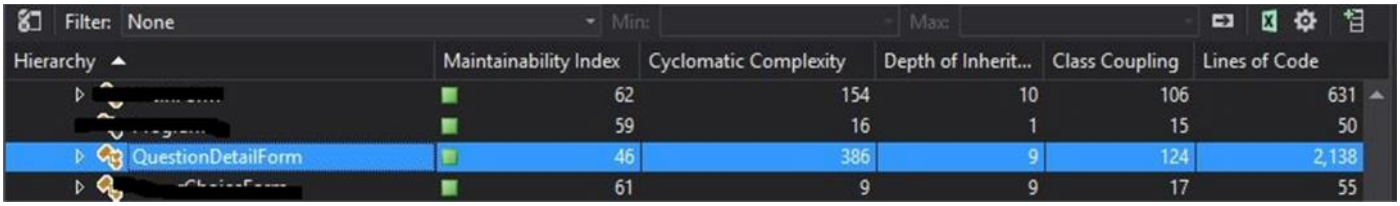

Figure 4: Question Detail Form

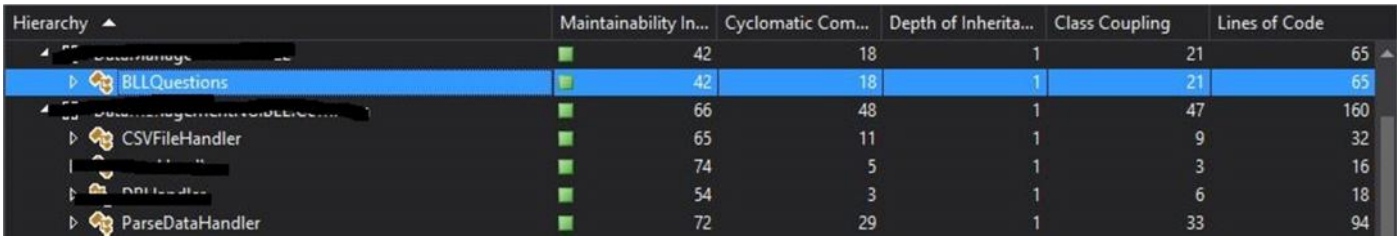

Figure 5: Parse Data Handler, CSV File Handler and BLL Questions